Uma comparação entre MapReduce e Tez para segmentação de imagens em ambientes de computação em nuvem

Patrick Nigri Happ [1]
Gilson Alexandre Ostwald Pedro da Costa [2]
Raul Queiroz Feitosa [1,2]

[1] Pontifícia Universidade Católica do Rio de Janeiro- PUC-Rio
Caixa Postal 38097 - 22451-900 – Rio de Janeiro - RJ, Brasil
{patrick, raul}@ele.puc-rio.br

[2] Universidade do Estado do Rio de Janeiro - UERJ
20550-900 - Rio de Janeiro - RJ, Brasil
{gilson.costa}@ime.uerj.br

Abstract. Driven mainly by the modern advances in the Earth Observation technology in the last years, the increase of the remote sensing data volume represents a new challenge. The current available image processing solutions fail to deliver the expected performance and scalability required to deal with this large volume of data. Aiming to face this problem, the authors proposed, in a recent work, a distributed strategy for region growing segmentation of arbitrarily large images. The presented strategy is able to perform in cloud-computing environments and most of the distributed architectures. The original implementation is based on the MapReduce model, which offers a highly scalable and reliable framework for storing and processing massive data in cloud computing environments. However, MapReduce is losing popularity lately and it is being slowly replaced by different engines that have been emerged. Since the distributed image segmentation is a method independent from its implementation, this paper aim to compare the original implementation using MapReduce to a new implementation using a different distributed framework. In this work, the new implementation is based on Apache Tez. Tez enhances the MapReduce paradigm by improving its speed while maintaining MapReduce's ability to scale to petabytes of data. The experiments carried out on a virtual cluster in a commercial cloud-computing infrastructure demonstrated that both implementations present a potential scalable and efficient solution, with Tez achieving a better performance.

Palavras-chave: remote sensing, image segmentation, distributed processing, cloud computing, sensoriamento remoto, segmentação de imagens, processamento distribuído, computação em nuvem.

1. Introdução

A análise de dados de sensoriamento é de extrema importância para os pesquisadores em diversas áreas de aplicação, como agricultura, resposta a desastres, planejamento urbano e operações de mineração, entre outras. Os avanços recentes nas tecnologias de observação da Terra, bem como o aumento do número de sistemas aéreos e orbitais em atividade têm fornecido uma maior quantidade de insumo para este fim. Exemplos ilustrativos são o Sentinel-1 da ESA que produz em torno de 1,5 GB por dia (Grabak, 2014) e o projeto EOSDIS da NASA gera cerca de 16 TB por dia (NASA Earthdata, 2015).

Adicionalmente, a redução considerável dos custos de aquisição de imagens junto a empresas privadas e o crescimento de conjuntos de dados de livre acesso difundidos nos últimos anos também contribuíram para o acesso a um volume maior de dados. Este cenário é pródigo em informações estratégicas que podem ser usadas pelos respectivos tomadores de decisão. Lidar com estes grandes volumes de dados, no entanto, representa um desafio principalmente quando estas informações devem ser obtidas em um curto prazo.

De fato, este cenário revela grandes desafios para a comunidade de sensoriamento remoto relacionados principalmente à capacidade de processamento desse grande volume de dados (Lee et al., 2015). Há, portanto, uma demanda por ferramentas automáticas para interpretação de imagens altamente escaláveis. Neste sentido, a computação distribuída é uma alternativa

comumente adotada quando se faz necessário o processamento de grandes conjuntos de dados.

De forma mais específica, a computação em nuvem é uma tendência, pois pode fornecer uma infraestrutura escalável para suportar diferentes necessidades de processamento (Fernández et al., 2014). Além disso, a existência de diferentes provedores de infraestrutura de computação em nuvem que oferecem um modelo de *pay-per-use* proporciona grande poder de computação aos usuários ao mesmo tempo em que os livra de preocupações com a aquisição ou manutenção de um hardware complexo.

Em um recente trabalho (Happ et al., 2016) foram apresentadas estratégias para segmentação por crescimento de regiões utilizando computação em nuvem. Estas soluções possibilitam o processamento distribuído de imagens muito grandes em um cluster físico ou virtual. As estratégias são baseadas na divisão da imagem em *tiles* e utilizam um mecanismo de indexação específico junto a um método de costura hierárquica para suprimir os artefatos que podem ser gerados ao longo das bordas. Experimentos conduzidos em uma implementação baseada no paradigma de programação distribuída MapReduce (Dean e Ghemawa, 2008) demonstraram um elevado potencial de escalabilidade.

As estratégias mencionadas são independentes do modelo de programação adotado. Ao mesmo tempo, alternativas ao MapReduce em código aberto vêm sendo propostas, em particular o Apache Tez (Saha et al., 2015) tem atraído muita atenção recentemente. Trata-se de uma plataforma de código aberto para construção de processos dirigidos por fluxos de dados. O objetivo deste trabalho é comparar a eficiência computacional do MapReduce com o Apache Tez, tomando como base o algoritmo de segmentação distribuída supracitado.

Uma das vantagens de utilizar Tez é a sua compatibilidade e integração com outras soluções consagradas como Pig (Olston, 2008) e Hive (Thusoo, 2009). Além disso, Tez tem sido considerado superior ao MapReduce por utilizar o sistema distribuído de arquivos de forma mais efetiva e processar várias fases de *reduce* sem a necessidade de fases de *map*, entre outros benefícios (Singh, 2016).

O restante deste artigo está organizado da seguinte forma. Na próxima seção as estratégias de segmentação distribuída são descritas de forma breve. Na seção seguinte, define-se a arquitetura geral e uma implementação da solução apresentada. Na seção 4, apresentam-se a análise experimental e os resultados. Por fim, o trabalho é concluído na Seção 5 e possíveis trabalhos futuros são indicados.

2. Segmentação por Crescimento de Regiões na Nuvem

Entre as etapas da análise de imagens, a segmentação geralmente é associada a um alto custo computacional. Apesar da existência de diversas soluções paralelas com a finalidade de acelerar esta tarefa, o problema recai sobre a capacidade de processar imagens muito grandes.

A solução adotada neste artigo se baseia nas estratégias apresentadas por Happ et al. (2016), que consistem uma solução escalável e eficiente para processar imagens de grandes tamanhos em um ambiente em nuvem. Uma breve explicação dos métodos apresenta-se a seguir.

O crescimento de regiões é um processo iterativo constituído de verificação constante de vizinhança e possui fortes dependências entre segmentos adjacentes. A divisão de imagens em *tiles* é uma solução potencial que requer algum mecanismo específico para lidar com os segmentos localizados nas bordas dos *tiles*.

Em suma, a estratégia proposta por Happ et al. (2016) lida com o problema por meio de três etapas. Inicialmente a imagem é dividida em *tiles* de forma a gerar conjuntos de dados independentes passíveis de distribuição. Em seguida, o algoritmo de segmentação propriamente dito é executado em cada um dos *tiles* de forma independente. Por fim, um método de pós-processamento é utilizado para costurar os segmentos adjacentes que tocam as

bordas de diferentes *tiles* com a finalidade de suprimir os artefatos gerados pelas segmentações independentes.

Para o funcionamento correto destas etapas, pressupõe-se que os segmentos internos, os que não tocam nas bordas dos *tiles*, estão corretamente delineados, isto é, não sofrem com a divisão da imagem. Embora essa suposição não seja necessariamente verdadeira em todos os casos, ela faz sentido basicamente porque esses segmentos estão menos sujeitos à potencial influência de *pixels* em *tiles* adjacentes. Esta hipótese implica em uma quantidade muito menor de processamento durante o processo de costura, e permite que segmentos de *tiles* diferentes sejam agrupados sem problemas críticos de memória.

Além disso, um eficiente esquema de indexação espacial com diferentes níveis hierárquicos é utilizado para determinar as extensões geográficas dos *tiles* de imagem e rotular os segmentos. Este esquema de indexação suporta o agrupamento de segmentos numa etapa de pós-processamento de acordo com a sua localização espacial, de modo que segmentos adjacentes façam parte da mesma distribuição de dados no pós-processamento.

Três estratégias de pós-processamento são definidas no trabalho de Happ et al. (2016): Pós-Processamento Simples (SPP), Pós-Processamento Hierárquico (HPP) e Pós-Processamento Hierárquico com Re-segmentação (HPPR). A primeira é a mais rápida, porém produz artefatos no resultado. Já a última consome mais tempo, mas fornece resultados sem artefatos. Neste trabalho, o interesse recai sobre a primeira estratégia, a SPP, que é utilizada para a execução dos experimentos. A SPP possui um único passo de pós-processamento, mas que é suficiente para permitir que segmentos de *tiles* adjacentes sejam fundidos.

3. Definição da Arquitetura e Implementação

De maneira geral, pode-se definir uma arquitetura para a segmentação distribuída composta por três camadas em diferentes níveis de abstração (Figura 1). A primeira camada está relacionada com a definição das entradas e parâmetros de segmentação e configurações do ambiente distribuído. Por esta razão, é conhecida como camada de definição.

A segunda camada é chamada de camada de segmentação e é organizada em uma estrutura de alto nível a fim de esconder a complexidade da programação distribuída propriamente dita. Esta camada permite incluir novos algoritmos de segmentação e outros métodos desejados, além de alterar o fluxo de execução.

A terceira camada é a camada de distribuição. Por estar relacionada com o processamento distribuído de fato, somente programadores com habilidades de programação distribuída serão capazes de interagir com essa camada. A ligação entre as camadas de segmentação e de distribuição é definida por uma tradução, onde as instruções de alto nível são compiladas em código de processamento distribuído.

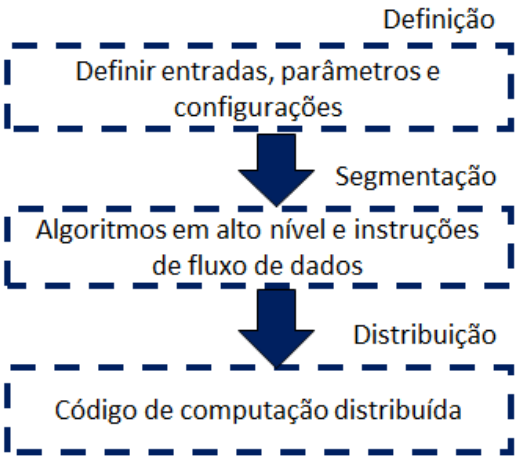

Figura 1. Arquitetura da segmentação distribuída.

3.1 Implementação Original com MapReduce

Tomando como base a arquitetura descrita, é possível implementar esta solução de diferentes formas. A camada de distribuição deve estar baseada em uma estrutura que suporte processamento e distribuição de dados. Neste caso, optou-se pelo MapReduce, um dos modelos de programação mais populares para o processamento de grandes conjuntos de dados nuvem (Assunção et al., 2015), e sua implementação de código aberto no Apache Hadoop. Hadoop (Apache Hadoop, 2016) é uma estrutura amplamente utilizada para o processamento distribuído de grandes conjuntos de dados entre clusters de computadores. O Hadoop é baseado em dois componentes principais: o sistema de arquivos distribuídos (HDFS) e o modelo de programação MapReduce. O HDFS é projetado e otimizado para alto desempenho de processamento e funciona melhor com arquivos grandes (gigabytes ou maiores). Já o Hadoop MapReduce é um modelo de programação baseado em um simples paradigma de processamento de dados composto de três fases principais: map, shuffle e reduce (White, 2010).

Em relação à camada de segmentação, é necessária uma plataforma intermediária que realize a interface com a camada de distribuição em alto nível. Neste caso, empregou-se a plataforma Pig, que é composta por uma linguagem de alto nível para expressar fluxos de dados (Pig Latin), e por um mecanismo que os compila em tarefas de MapReduce. O Pig Latin não só fornece uma maneira mais fácil de trabalhar com o modelo MapReduce distribuído como oferece uma capacidade de extensão de bibliotecas externas e métodos através das Funções Definidas pelo Usuário (UDFs).

A implementação do algoritmo de crescimento de regiões também é feita nesta camada e é baseada na proposta de Baatz e Schäpe (2000), que é amplamente utilizada em aplicações de sensoriamento remoto baseadas em objetos. O algoritmo considera cada *pixel* como uma semente, isto é, como um segmento inicial; e utiliza critérios de homogeneidade espectral e morfológica na decisão de fusão entre regiões. O algoritmo é codificado em Java e estruturado como uma UDF do Pig a fim de ser invocado por um script Pig Latin. Algoritmos adicionais podem ser incorporados seguindo a mesma estrutura.

Por fim, a camada de definição está associada às ferramentas disponíveis pelo serviço que instancia o cluster na nuvem, bem como a definição das imagens e parâmetros de segmentação.

3.2 Implementação Alternativa com Tez

Dado que as camadas abstratas presentes na arquitetura são relativamente independentes, é possível alterar uma das camadas sem afetar o restante da implementação. Neste caso, é possível manter a implementação original para as camadas de definição e segmentação e modificar apenas a camada de distribuição.

Assim, ao invés do script Pig Latin traduzir as tarefas para MapReduce, a tradução será feita para o fluxo de processamento definido pelo Tez, o qual será responsável por executar o processamento distribuído e gerenciar o HDFS. Tez visa à construção de uma plataforma de aplicação que permite complexos grafos direcionados acíclicos (DAG) de tarefas para o processamento de dados. Tez aprimora o paradigma MapReduce, melhorando a sua velocidade enquanto mantém a capacidade do MapReduce em escalar petabytes de dados. Tez possui rotinas para definições de fluxos de dados com simples implantação e um gerenciamento de recursos otimizado (Apache Tez, 2016). Além disso, o processamento de dados, que poderia ser transformado em diversas tarefas de MapReduce, pode vir a ser executado em uma única tarefa Tez.

4. Análise Experimental

Para avaliar o desempenho das diferentes implementações foram realizados experimentos em um *cluster* virtual utilizando uma infraestrutura comercial de computação em nuvem. Foi utilizada uma imagem com subconjuntos de diferentes tamanhos, em *clusters* com diferentes quantidades de nós, executando as duas implementações: MapReduce e Tez. Dentre as três estratégias de pós-processamento, a SPP, que é mais rápida, foi selecionada. A seguir são fornecidos maiores detalhes sobre o ambiente de execução, as imagens de teste e os resultados obtidos.

4.1 Ambiente em Nuvem

Os experimentos foram realizados utilizando a infraestrutura fornecida pela Amazon Web Services (AWS), uma provedora comercial de infraestrutura de computação. O Amazon Simple Storage Service (S3) foi usado para armazenar os dados de entrada e saída, bem como os programas e bibliotecas necessários. O serviço Amazon Elastic MapReduce (EMR) foi usado para gerenciar a plataforma do Hadoop em um cluster virtual dinamicamente construído por instâncias do Amazon Elastic Compute Cloud (EC2).

Os experimentos foram executados em máquinas do tipo m3.xlarge, em clusters variando entre 2, 4 e 8 nós, além de um nó adicional para agendar e gerenciar as tarefas. Cada nó possui um processador Intel Xeon E5-2670 v2 com quatro núcleos, operando a 2.5GHz, com 15 GB de RAM e dois discos SSD de 40GB. Para a configuração de software dos nós utilizados, foi selecionada a versão emr-5.1.0 contendo Hadoop 2.7.3 e Pig 0.16. No caso dos experimentos utilizando Tez, foi também incluído Tez 0.8.4. As principais configurações adicionais envolvem a definição de uma maior alocação de memória em substituição ao valor padrão.

4.2 Imagens de Teste

Uma cena WorldView-2 obtida em 2012 foi utilizada para a execução dos experimentos. A cena cobre áreas rurais e urbanas do município de São José dos Campos no estado de São Paulo (Figura 2). A figura encontra-se rotacionada para facilitar a visualização.

Figura 2. Cena WorldView-2 selecionada para os experimentos. A imagem encontra-se rotacionada para facilitar a visualização.

A imagem completa, denominada de imagem 32k, possui 22740 × 32000 *pixels* e é resultante de uma fusão pelo método IHS das bandas multiespectrais (5, 3 e 2) correspondentes a *red*, *green* e *blue*, com 2m de resolução espacial, com a banda pancromática com 0,5m de resolução. Para verificar o desempenho com imagens de diferentes tamanhos, foram utilizados outros três subconjuntos da imagem mantendo sempre o centro como origem. Os subconjuntos possuem tamanhos 11370 × 16000, 5685 × 8000 e 2844 × 4000 *pixels* e são denominados, respectivamente, de 16K, 8K, 4K.

O tamanho dos tiles foi definido em 1024 × 1024 *pixels*. Desta forma a imagem de 32k possui 736 tiles, enquanto as de 16k, 8k e 4k possuem, respectivamente, 192, 56 e 16 tiles.

4.3 Resultados

A figura 3, de (a) a (d), demonstra o tempo de execução de cada implementação para as imagens de 4K, 8K, 16K e 32K, respectivamente. Claramente, o tempo de execução é reduzido quando há um aumento na quantidade de nós do cluster. Da mesma forma, verifica-se que imagens maiores possuem um maior tempo de execução.

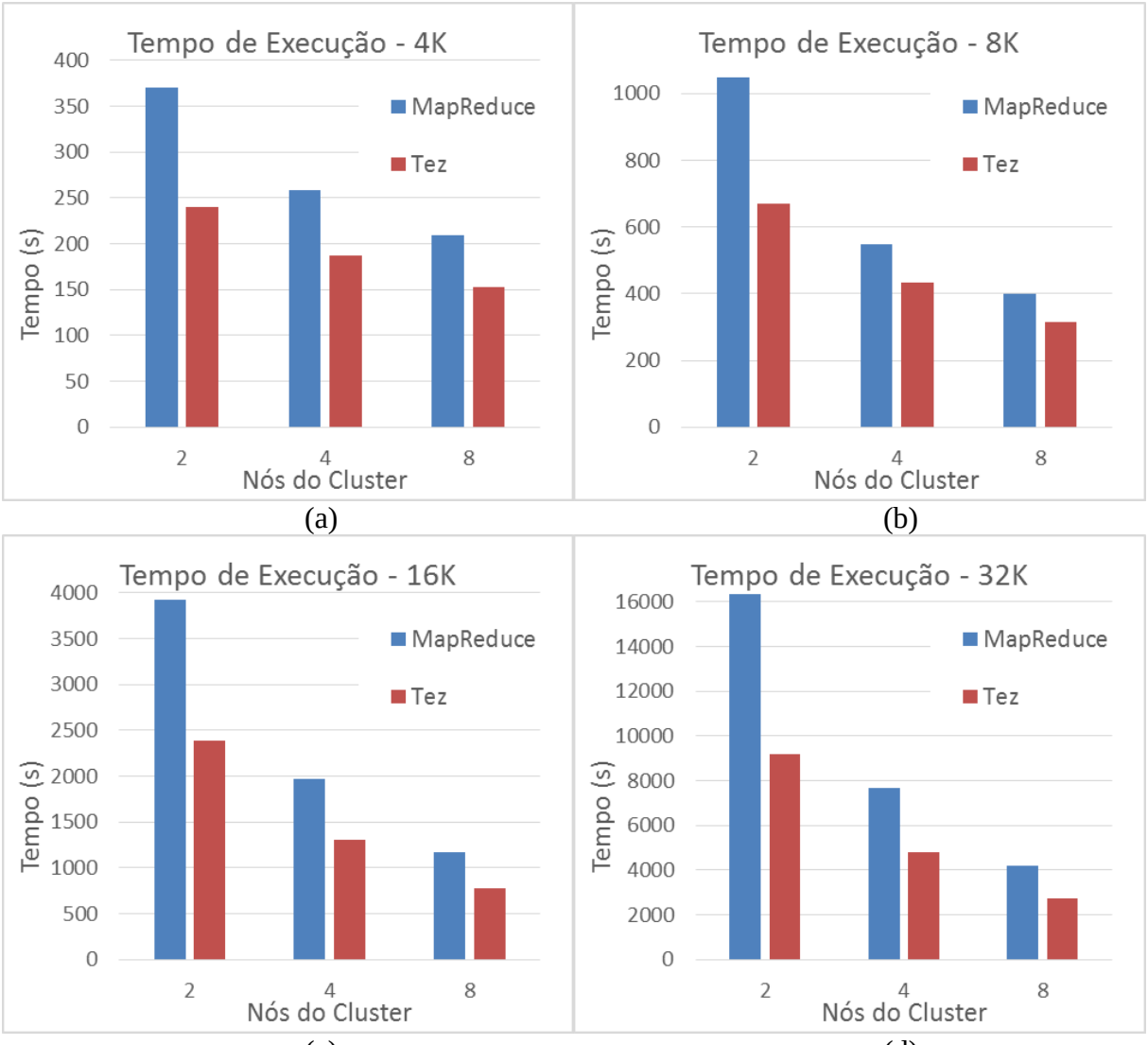

Figura 3. Tempos de execução da estratégia SPP implementada com MapReduce e Tez utilizando 2, 4 e 8 nós do cluster e as imagens de (a) 4K, (b) 8K, (c) 16K e (d) 32K.

A observação mais importante neste caso diz respeito à comparação entre as implementações MapReduce e Tez. Verifica-se que para todos os casos a implementação em Tez é mais rápida do que a implementaçãos em MapReduce. A Figura 4 apresenta a aceleração obtida ao comparar as duas implementações com a finalidade de auxiliar esta análise.

Na média, a aceleração do Tez em relação ao MapReduce foi de 1,5. O melhor caso foi na execução da imagem de 32K com 2 nós, onde alcançou-se uma aceleração de 1,78. O pior caso foi a aceleração de 1,27 para a imagem de 8K com 8 nós. Nota-se ainda que, para cada uma das imagens, o uso de apenas 2 nós resultou em uma maior aceleração, enquanto o uso de 4 e 8 nós resultou em uma aceleração similar.

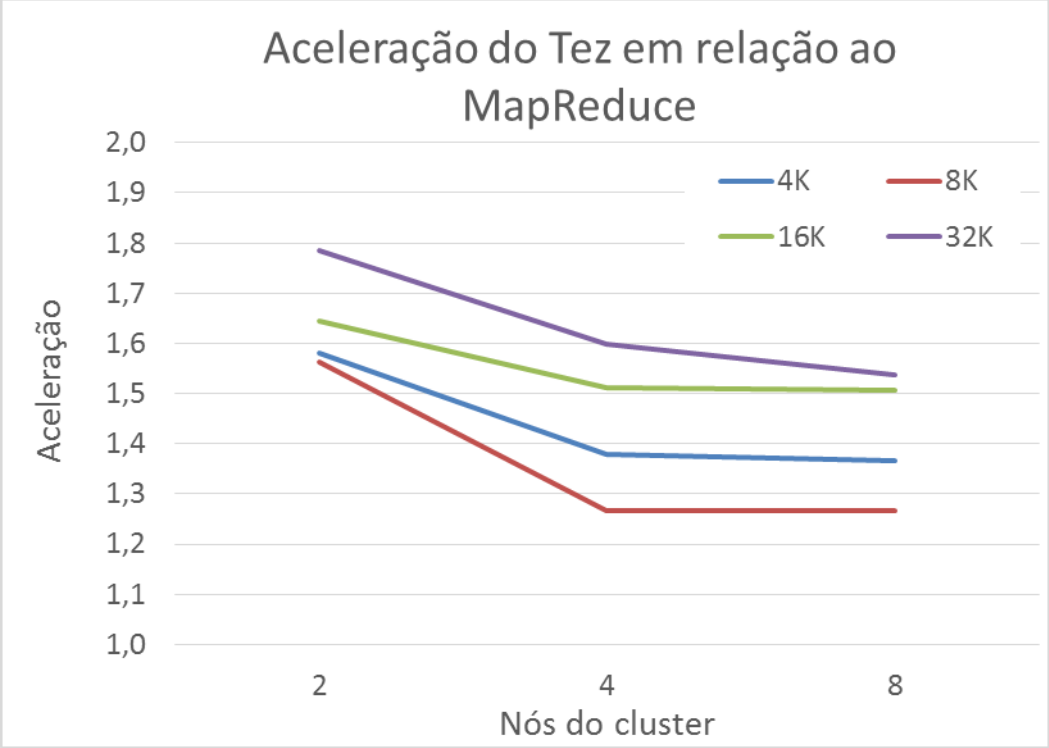

Figura 4. Acelerações da implementação em Tez em relação à implementação em MapReduce utilizando 2, 4 e 8 nós do cluster para as imagens de 4K, 8K, 16K E 32K.

5. Conclusões

Este trabalho apresenta uma estratégia de segmentação de regiões distribuída capaz de lidar com imagens arbitrariamente grandes em cluster de computadores, que podem ser estruturadas em um ambiente em nuvem. Mais especificamente, este artigo demonstra que diferentes implementações distribuídas podem ser adotadas, mantendo o potencial de escalabilidade e eficiência da solução.

Experimentos utilizando um ambiente comercial para computação em nuvem demonstram que a implementação original utilizando MapReduce pode ser substituída por outras novas plataformas que sugerem acelerar o processamento. Neste trabalho avaliou-se o Apache Tez que se mostrou mais eficiente, obtendo um maior ganho de desempenho computacional.

Como trabalhos futuros vislumbram-se: verificar o desempenho das outras estratégias de pós-processamento não contempladas neste trabalho; e a realização de experimentos com outras plataformas de programação distribuída disponíveis, tais como o Spark.

Agradecimentos

Os autores agradecem o financiamento fornecido pela CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior).

Referências Bibliográficas

Apache Hadoop. Disponível em: <http://hadoop.apache.org/> Acesso em: 10.nov.2016.

Apache Tez. Disponível em <https://tez.apache.org/> Acesso em: 10.nov.2016.

Assunção, M. D.; Calheiros, R. N., Bianchi, S., Netto, M. A. S., Buyya, R. Big Data computing and clouds: Trends and future directions. Journal of Parallel and Distributed Computing, vol. 79–80, pages 3-15, 2015.

Baatz, M.; Schäpe, A. Multiresolution Segmentation: an optimization approach for high quality multi-scale image segmentation. In: Angewandte Geographische Informationsverarbeitung XII, Heidelberg, 2000.

Dean, J.; Ghemawa, S., MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, v. 51, n. 1, p. 107-113, 2008.

Fernández, A.; del Río, S.; López, V.; Bawakid, A.; del Jesus, M. J.; Benítez, J. M.; Herrera, F. Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 4, no. 5, pp. 380-409, September/October 2014.

Grabak, P. Sentinel-1 Mission Status. In: 15th Meeting of the International Ice Charting Working Group (IICWG), 2014.

Happ, P. N.; da Costa, G. A. O. P.; Bentes, C.; Feitosa, R. Q.; Ferreira, R. S.; Farias, R. A Cloud Computing Strategy for Region-Growing Segmentation. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol.PP, no.99, pp.1-10, 2016.

Lee, J.; Kang, M. Geospatial Big Data: Challenges and Opportunities. Big Data Research, vol. 2, pp. 74-81, June 2015.

NASA Earthdata. The Earth Observing System Data and Information System. Disponível em: <https://earthdata.nasa.gov/> Acesso em: 10.nov.2016.

Olston, C.; Reed, B.; Srivastava, U.; Kumar, R.; Tomkins, A. Pig latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD '08, pages 1099–1110, 2008.

Saha, B.; Shah, H.; Seth, S.; Vijayaraghavan, G.; Murthy, M.; Curino. C. Apache Tez: A Unifying Framework for Modeling and Building Data Processing Applications. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15), pages 1357-1369, 2015.

Singh, R.; Kaur, P. J. Analyzing performance of Apache Tez and MapReduce with hadoop multinode cluster on Amazon cloud. Journal of Big Data, vol. 3, no 1, 2016.

Thusoo, A.; Sarma, J. S.; Jain, N.; Shao, Z.; Chakka, P.; Anthony, S.; Liu, H.; Wyckoff, P.; Murthy, R. Hive – a warehousing solution over a map-reduce framework. In: PVLDB, 2009.

White, T. Hadoop: The Definitive Guide, 2nd ed.: O'Reilly Media / Yahoo Press, 2010.